MCS® 96 Microcontrollers — The Perfect Match for Fuzzy Logic Applications

Mohammed Fennich
Applications Engineer
Intel Corporation
Article ID# 0704

In the domain of control systems, fuzzy logic proved very useful and very suitable for systems without mathematical models and systems with too much uncertainty. In a way, it came to the rescue when conventional methods started to reach dead ends.

However, fuzzy logic also has its down side. The major drawback of the emerging technology is the lack of a formal design methodology. The other disadvantage is that the resulting system is not analytical, and therefore any mathematical analysis on paper is impossible with current methods.

The lack of formalism at the application level, however, did not obstruct the continuous path of fuzzy logic. Even with this handicap, a large number of applications have been implemented using fuzzy controllers. The basic and essential requirements were found to be of a hardware and software nature.

When a conventional embedded microcontroller is used in conjunction with a software fuzzy logic algorithm, the rules that constitute the base of the algorithm are evaluated in a sequence, one after the other. Once all the rules are evaluated, their outputs are combined to provide a single value that will be defuzzified.

In contrast, if a dedicated fuzzy processor is used, the rules are evaluated in parallel. The parallel processing method suggests a fast processing cycle. However, in this case, data acquisition and data output still have to be done using conventional peripherals. The time gained in processing the rules in parallel can be lost in acquiring the data via external peripherals.

For most control systems, the best solution is to use a software fuzzy algorithm on a microcontroller with fast internal peripherals. In this case, the sequential rule-processing scheme is transparent to the user and the process seems to have been done in parallel.

The MCS® 96 microcontrollers are equipped with high-performance internal peripherals that make data acquisition, conditioning, and output fast and easy to handle. These peripherals, the wide range of addressing modes, and the powerful set of instructions that the family offers make these controllers very suitable for fuzzy logic applications.

The example below shows how some of the peripherals of the 80C196KD were used to acquire data via sensors in a closed-loop system with two inputs and one output. An overview of the hardware involved is given, and the steps taken in order to develop the software are also presented.

An experimental model representing a container crane was constructed. The system built consists of a pendulum of mass M and length L. The pendulum is hooked to a car that runs on rails. When the car moves from right to left or from left to right, the pendulum moves with it and swings.

The objective is to use fuzzy control to move the car from an initial position to a pre-assigned final position, and at the same time limit the oscillations of the pendulum. The block diagram of the system is shown in Figure 1.

An optical quadrature encoder is used as a position sensor. It has two outputs, channel A and channel B.

The sensor was mounted on the shaft of the motor. When the shaft turns, the sensor outputs the two square

Continued on page 10

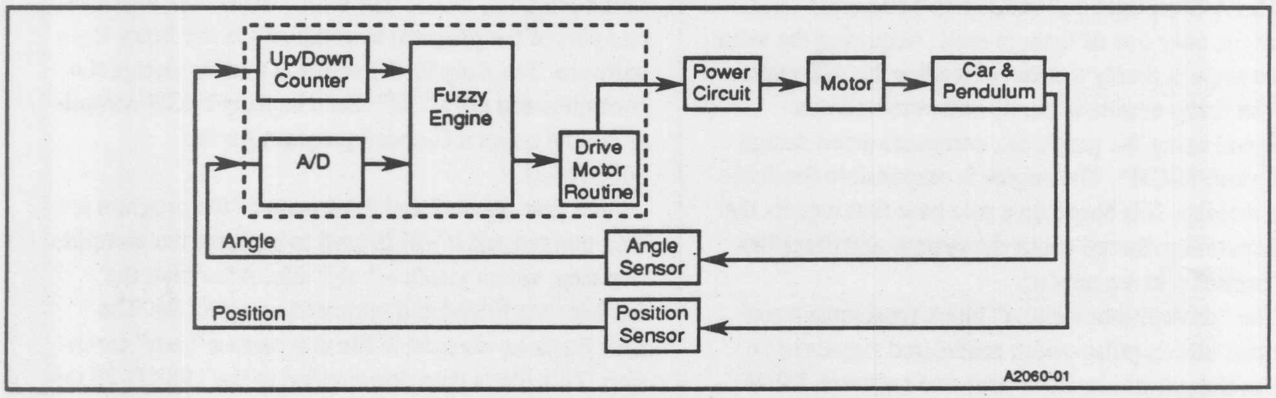

Figure 1. Block Diagram of the System

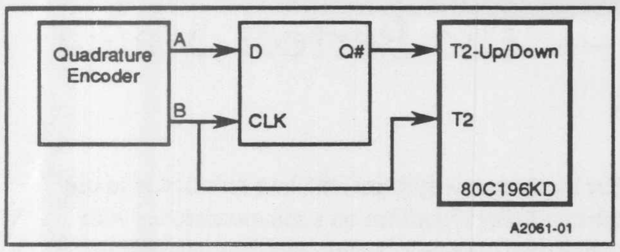

Figure 2. Connection of the Position Sensor to the 80C196KD

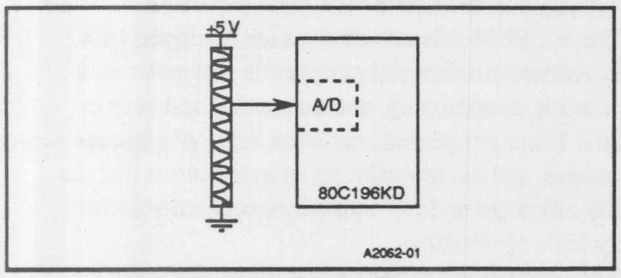

Figure 3. Angle Sensor Connections

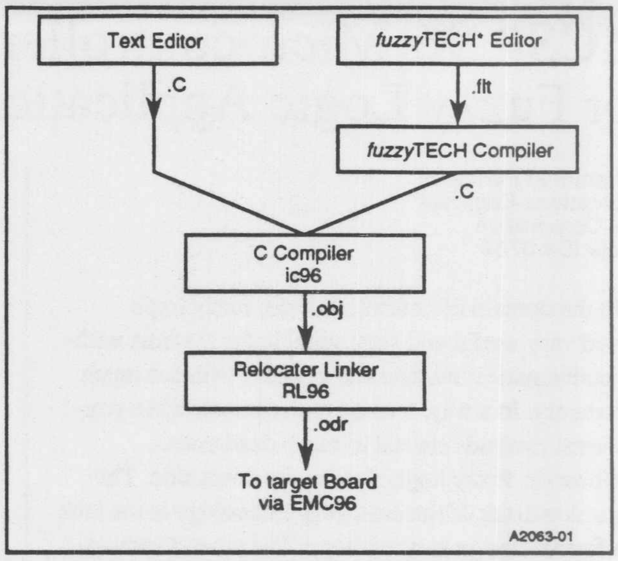

Figure 4. Program development

Continued from page 9

wave signals. To determine in what direction the shaft is turning, a D flip-flop was used. It outputs a signal that is either high or low, depending on the direction. The electrical connections are shown in Figure 2.

To acquire the distance information, the pulses out of channel B are fed to timer 2 on the 80C196KD controller. Timer 2 is used in the up-down mode. When the T2-up/down pin of the 80C196KD is low, timer 2 counts up; when it is high, timer 2 counts down. Note that the T2-up/down pin is fed the signal from the Q# output of the flip-flop, which allows timer 2 to count up or down depending on the direction.

The sensor used to acquire angle values is simply a potentiometer powered between zero and five volts. When the pendulum swings, it causes the shaft of the sensor to turn. In other words, it moves the location of the cursor of the potentiometer.

The electrical connections between the sensor and the 80C196KD can be represented as Figure 3 shows.

The A/D on the 80C196KD has 8 channels. In this example, only one of them is used. Acquiring the value of the angle is simply a matter of reading the A/D register.

The fuzzy engine is purely software and was designed using the graphical, computer-aided design tool, *fuzzy*TECH*. The engine is responsible for decision making. It is based on a rule base that depicts the information gathered about the system and describes the control scheme needed.

The "drive-motor routine" block represents a routine that allows pulse-width modulated signals to be sent to the power circuit. It uses two registers, PWM1-control and PWM2-control, of the 80C196KD. Writing a number between zero and 255 to these registers causes the pins, PWM1 and PWM2, to output pulse trains of which the duty cycle is dependent on the value that's in the PWMx-control register. The hardware connections were made so that PWM1 drives the car-pendulum system to the left and PWM2 drives it to the right.

The power circuit block is an H-bridge built using discrete components. The motor used is a simple DC motor used in this project at 6.5 volts with a maximum current of 0.4 amps.

The software side of this project consists of an executable program that is made of three major parts: the data acquisition section, the fuzzy controller, and the data ouput section. The program is developed in a PC environment and then downloaded to the 196KD-20 target board on which the 80C196KD resides. (This same target board is supplied with the Project Builder 196 kits.) In order to obtain the executable program, the steps summarized in Figure 4 were taken.

A text editor is used to write the first and third parts of the program, which take a ".C" extension. The second part of the program is obtained via the fuzzy logic software. The *fuzzy*TECH editor is used to design the controller and get a ".ftl" file. The *fuzzy*TECH compiler is used to get a C-coded program for the 80C196KD.

The first, second, and third parts of the program are then merged and iC-96 is used to perform the compilation step, which yields a ".obj" file. After that, the modules are linked and relocated using RL96. The final file is an executable file that takes a ".odr" extension. This file is then downloaded to the 196KD-20 target board using the communication software, ECM96.